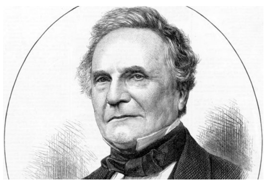
From 1821 -

It was all Babbage's Idea !

"..to calculate by steam"

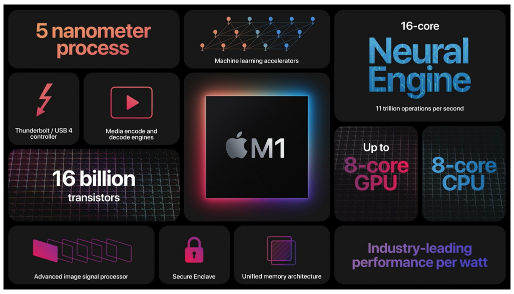


To 2021

- the Apple

M1

System on Chip


5 nanometer process
Machine learning accelerators
16-core
Neural Engine
11 trillion operations per second
Thunderbolt / USB 4 controller
Media encode and decode engines
M1
16 billion transistors
Up to 8-core GPU
8-core CPU
Advanced image signal processor
Secure Enclave
Unified memory architecture
Industry-leading performance per watt

This extends a talk which I originally gave 10 years ago. I've changed it, refreshed it and updated it a good deal.

It is 200 years since 1821, which could be regarded as when the idea of computers was born. At that date, mathematical tables existed, produced by tedious hand calculation, and full of errors. Babbage expressed the wish to "calculate by steam" and two years later he had built the first experimental difference engine. I am not going to spend time on that, because the story of computers only becomes interesting from the mid 20th century.

They <u>should</u> be as famous as Bill Gates !

*… these **Turing Award** Winners*

Wilkes, Corbato, Dijkstra, Wirth

Hoare, Richie & Thomson, Kay, Thacker

…. and

Lovelace Medal

Winners

Linus Torvalds

Sir Tim Berners-Lee OM KBE FRS

Steve Furber CBE, FRS

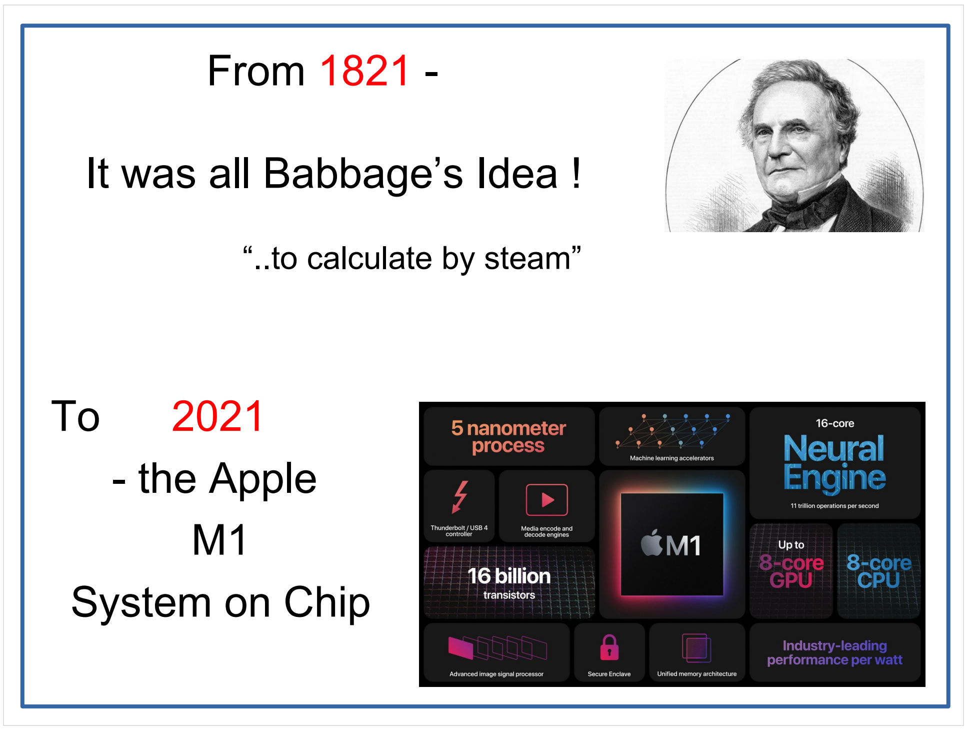
If we asked people to list the names of important people in the modern story of computers, most would find it hard to think of anyone except Bill Gates. In fact there are a large number of people who really deserve to be better known than Bill Gates!

Their scientific contributions have been much greater than his, though he is the one who has made his fortune! Many, have been recognised by prestigious awards such as the Turing Award and the Lovelace Medal.

Turing Awards are given by the American Association for Computing Machinery and include an award of $250,000. Some people refer to them as computing's Nobel Prizes.   Here are just a few recipients, particularly relevant to the things I will cover. today

Lovelace Medals, just as prestigious, are given by the British Computer Society – no money of course. Only about 20 have been given, and here are three recipients.

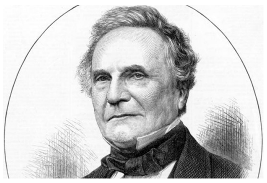
Of course we can't forget Bill Gates. But he has had no awards of this kind, though he has been recognised by honorary doctorates, an honorary knighthood, the Order of the Aztec Eagle etc.



From 1821 -

It was all Babbage's Idea !

"..to calculate by steam"

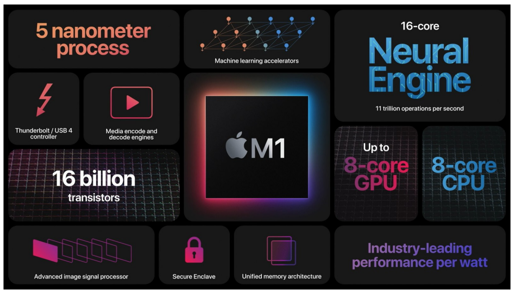


To 2021

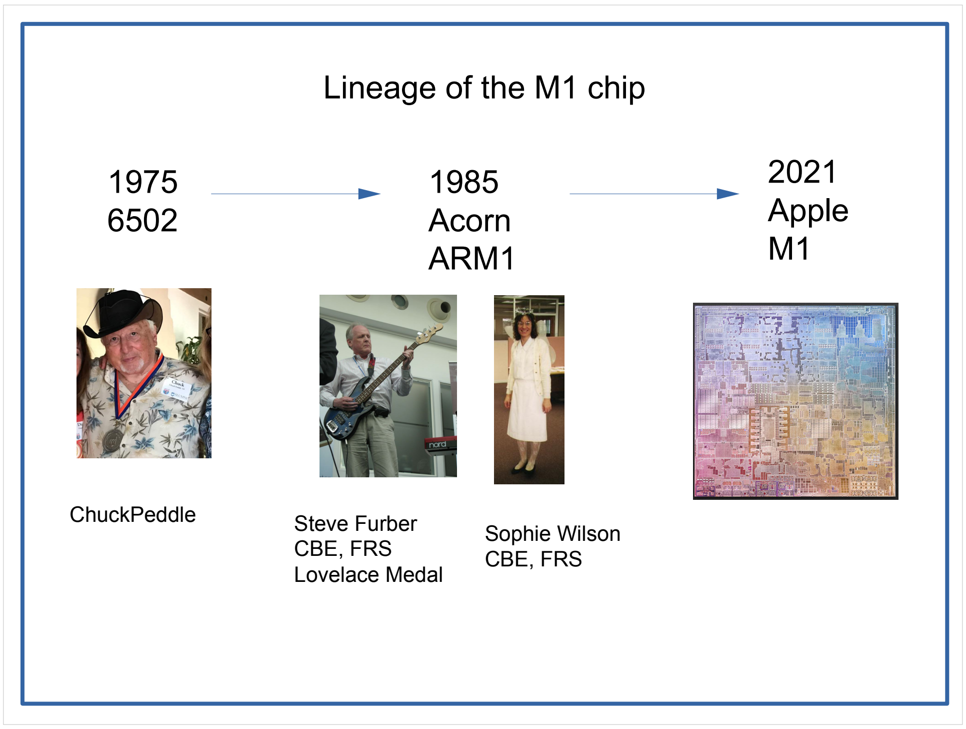
- the Apple

M1

System on Chip


5 nanometer process
Machine learning accelerators
16-core
Neural Engine
11 trillion operations per second
Thunderbolt / USB 4 controller
Media encode and decode engines
M1
16 billion transistors
Up to 8-core GPU
8-core CPU
Advanced image signal processor
Secure Enclave
Unified memory architecture
Industry-leading performance per watt

Now 2020 saw the significant launch of a new range of more powerful Apple Mac computers based on the Apple M1 system-on-a-chip. Note the 16 billion transistors on a chip, an incomprehensible number. That is made possible by the state of the art 5 nanometre process.



Lineage of the M1 chip

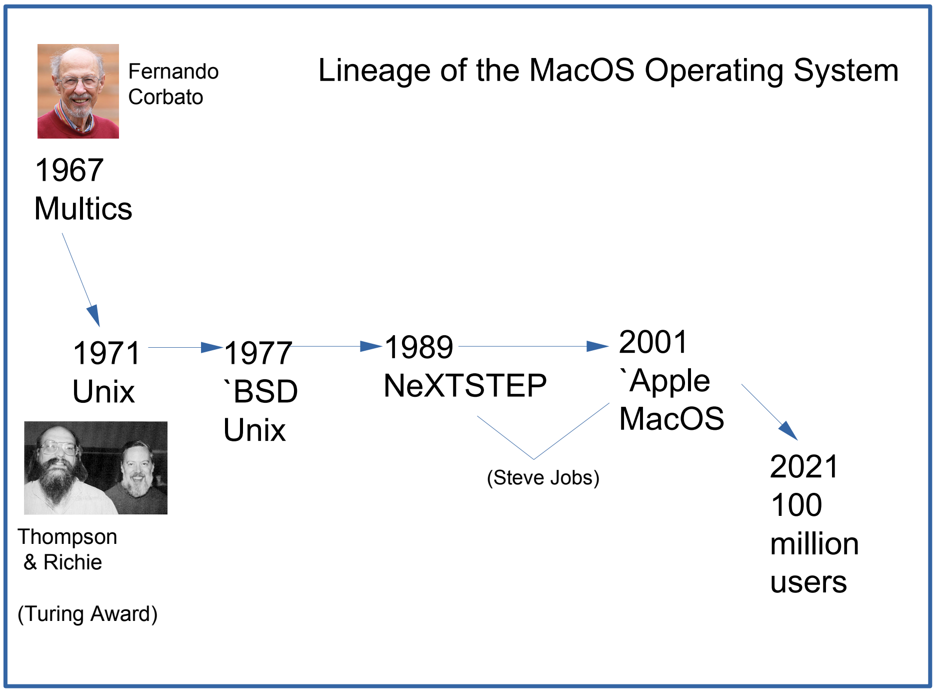
$$\begin{tikzcd}\text{1975 6502} \arrow[r] & \text{1985 Acorn ARM1} \arrow[r] & \text{2021 Apple M1}\end{tikzcd}$$

ChuckPeddle

Steve Furber
CBE, FRS
Lovelace Medal

Sophie Wilson
CBE, FRS

I will relate the technology to a succession of evolutionary events during over 50 years. In fact I will follow three main strands of that story as the theme of this talk. The first is the lineage of the extraordinary chip itself, going back almost 50 years to the work of Chuck Peddle in 1975, and then of Furber and Wilson in Cambridge in 1985. I will be enlarging upon this.



But the chip is no good without the software, and the second theme is the story of the software operating system of the range of computers, a story which starts in 1967 with Frederick Corbato, soon followed by the start of Unix 50 years ago,

Lineage of the 'SWIFT' Language

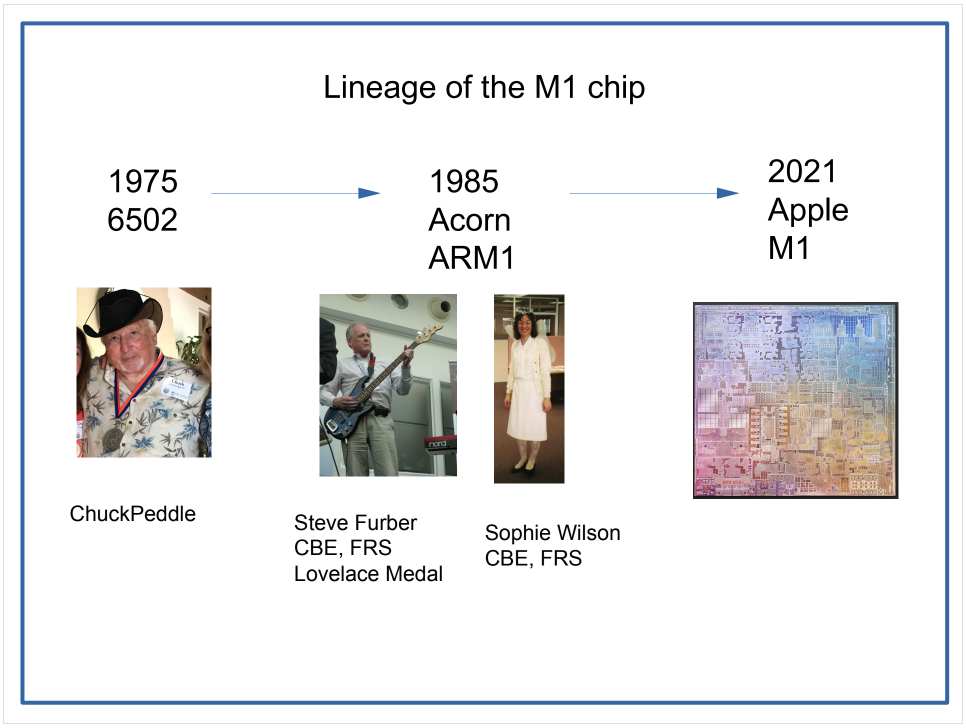
$$\begin{tikzcd}
\text{1960 ALGOL} \arrow[r] & \text{1978 'C'} \arrow[r] & \text{1988 Objective C} \arrow[r] & \text{2014 Swift}
\end{tikzcd}$$

Edsger Dijkstra
Turing Award

Denis Richie
Turing Award

Producing software requires computer languages and so my third theme is the story of the recent computer language, Swift, which is used to create the software. That story starts with Algol in 1960 followed in 1978 by the hugely important 'C'.

I will develop and explain these diagrams more fully, but it suffices for now to see that all the strands have origins a long time ago.   In all these strands progress has been evolutionary. It has to be, because backwards compatibility can rarely be sacrificed.



Lineage of the M1 chip

1975 6502 → 1985 Acorn ARM1 → 2021 Apple M1

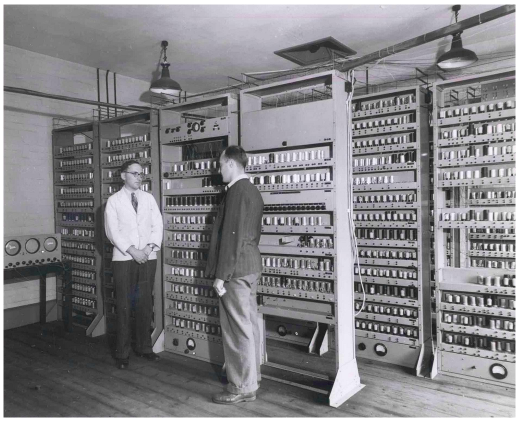
ChuckPeddle

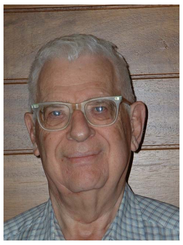
Steve Furber
CBE, FRS
Lovelace Medal

Sophie Wilson
CBE, FRS

My first strand leads to the Apple M1 chip. It is the history of computer hardware – the central processor and the associated devices.

EDSAC - 1949 and Leo 1 - 1951


Maurice Wilkes & EDSAC


David Wheeler

David Caminer

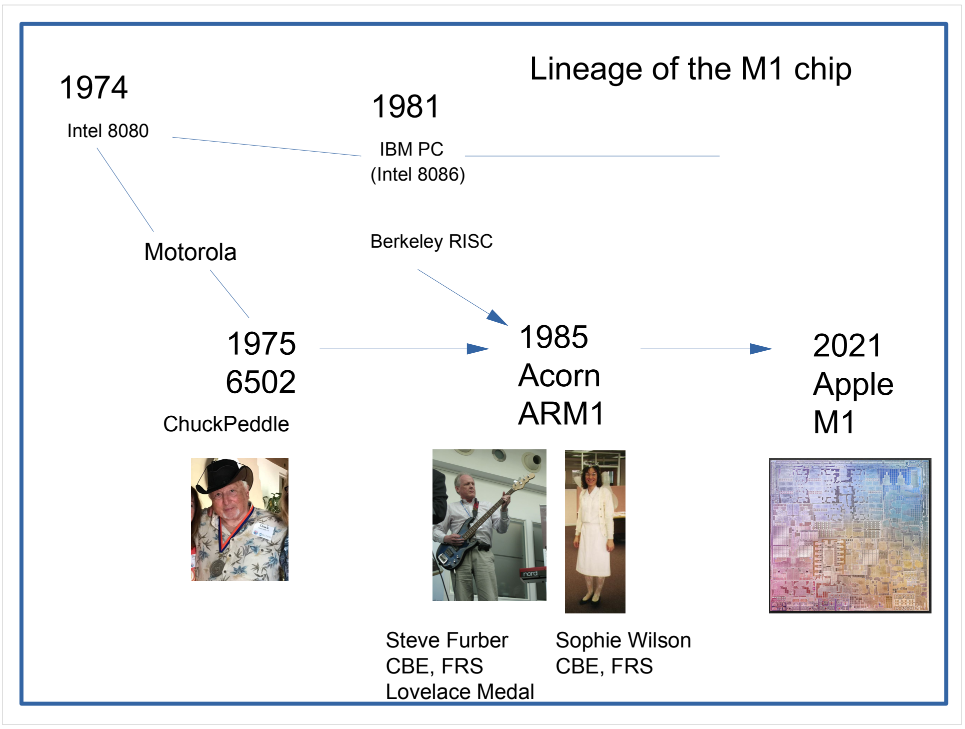
We start with EDSAC at Cambridge. The man who is generally reckoned to have won the race to produce a useful computer was Maurice Wilkes at Cambridge. EDSAC was switched on in 1949. That was actually the year I went to Cambridge and in 1952 I attended Maurice Wilkes lectures on computing and EDSAC.

EDSAC had more than 3000 valves. Memory was provided by tanks of mercury which implemented acoustic delay lines. At the National Physical Laboratory Turing was trying to build ACE with similar mercury delay lines. But EDSAC won the race to be the first computer. A mathematician called Tommy Gold, who lectured to me at Cambridge, had worked on radar during the war, and knew about mercury delay lines. His delay lines worked, while NPL and others were floundering.

So here is Wilkes with EDSAC. And here is David Wheeler, the post-grad who got the very first computer science PhD for his programming work on EDSAC. Their work can be said to be the foundation of Cambridge's silicon fen.

The next phase of the EDSAC story was, in many ways, even more remarkable. The Joe Lyons company, the tea shop chain, were already thinking about a business computer. They actually sent two people to America to investigate, and the Americans said that they should talk to Cambridge. In 1951, LEO 1, based on EDSAC, was built and was successfully used for payroll, inventory and production control tasks. And the met office also had a LEO 1. How did they do it in a machine with only 8000 bytes of memory. My laptop has 100,000 times as much memory! Here's David Caminer who did it. (One leg, lack of recognition apart from an Honorary Doctorate)

I must now move rapidly on. In the mid 1950s transistors quickly took over from valves, and magnetic ferrite cores were introduced for data storage. Processors were built from discrete components, And IBM was becoming a major computer manufacturer delivering their first computer in 1954. Computers had arrived.



The next big development came in 1971, with Intel's first microprocessor, followed by the 8080 in 1974. It took a few years more before Intel's microprocessor business really exploded with the 8086 processor, and it's adoption in the IBM PC, launched in 1981. That was a milestone. The PC was a modular open architecture. Any manufacturer, or any individual, could buy a self-selected collection of bits and pieces and build a working desktop computer. I built several, about 30 years ago. I bought the parts from Novatech – and you can still go to Novatech and do the same.

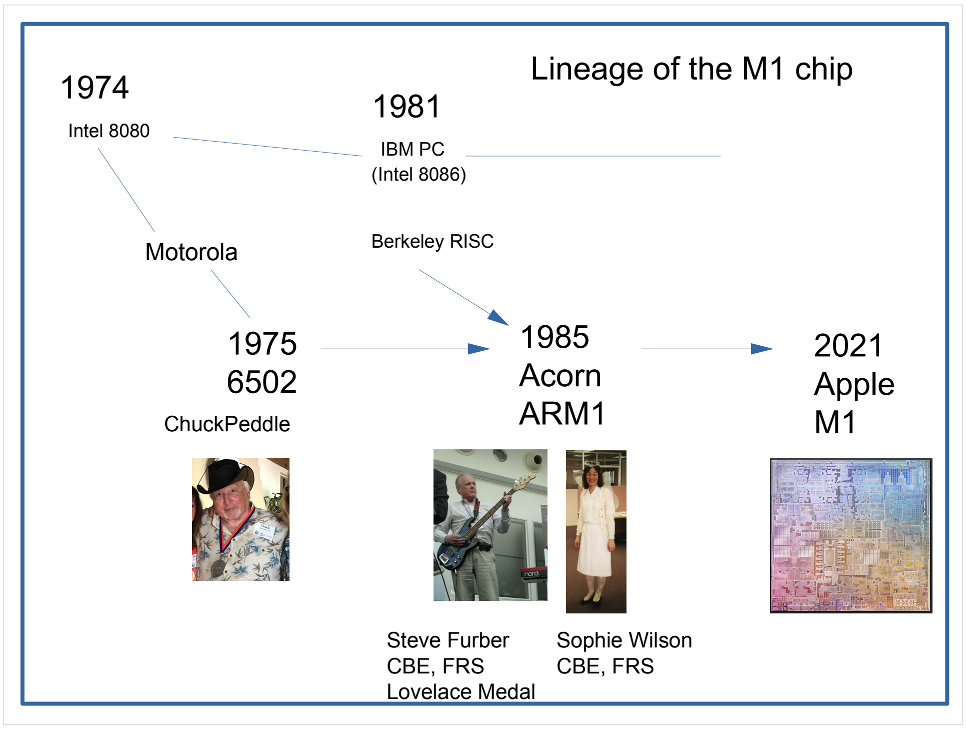
Intel flourished – but their approach was to build in more and more complexity into ever more powerful processors. The instruction set got ever bigger. That required variable length instructions which actually became a handicap.

But seeds were soon sown elsewhere which would eventually end Intel's dominance and lead to the Apple M1 System-on-a-chip.

Let's go back to the microprocessor CPU of the 1970s. The U.S. company Motorola had been quick to produce its own successful line of microprocessors, matching Intel's prices of some 360 dollars for a single microprocessor.

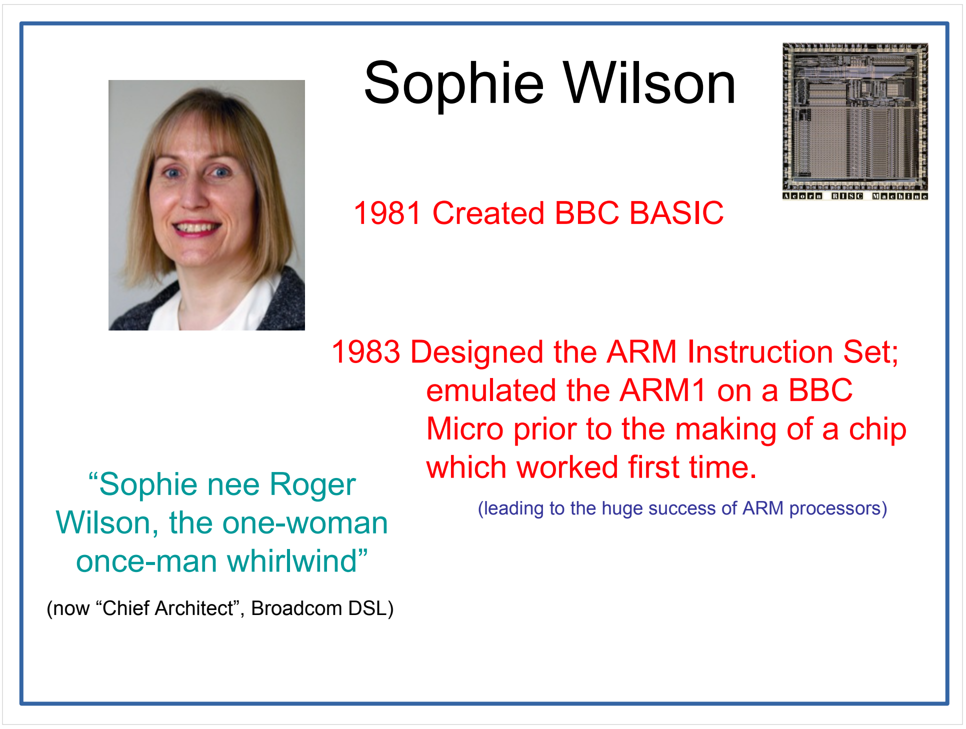
Chuck Peddle was at Motorola in 1973. He conceived a 30 dollar microprocessor taking a simpler approach . Motorola management would not support him. So he left, with several colleagues, and created the 6502. And based on the 6502 he launched the Commodore PET. And the 6502 rapidly became the heart of other home computers such as the Apple 2 and Acorn BBC Micro of 1981. Acorn were soon thinking "What next" and realised that they would need a more powerful microprocessor. Nothing from Intel or others met their reqirements. So Steve Furber and Sophie Wilson were charged with finding an answer. Later, Herman Hauser said

“When we decided to do a microprocessor on our own, I gave Steve Furber and Sophie Wilson two things which National, Intel and Motorola had never given their design teams: the first was no money; the second was no people. The only way they could do it was to keep it really simple." (Herman Hauser, Acorn)

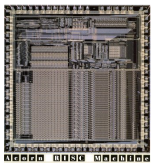


Sounded impossible. But the 6502 processor provided the lead. They visited the “Western Design Center” in Phoenix, where 6502 related design work was done. It was a revelation. There were a couple of engineers and some college kids working in a modest bungalow. Furber and Wilson were convinced that designing a simple processor was well within their capabilities.

This was late 1983. Sophie Wilson plunged right in. Besides her knowledge of the 6502 she was persuaded by ideas from the 1981 Berkeley RISC project -an important product of the University of California, Berkeley. RISC is “Reduced Instruction Set Computing” where the processor has quite a small set of equal length instructions.



Sophie Wilson



1981 Created BBC BASIC

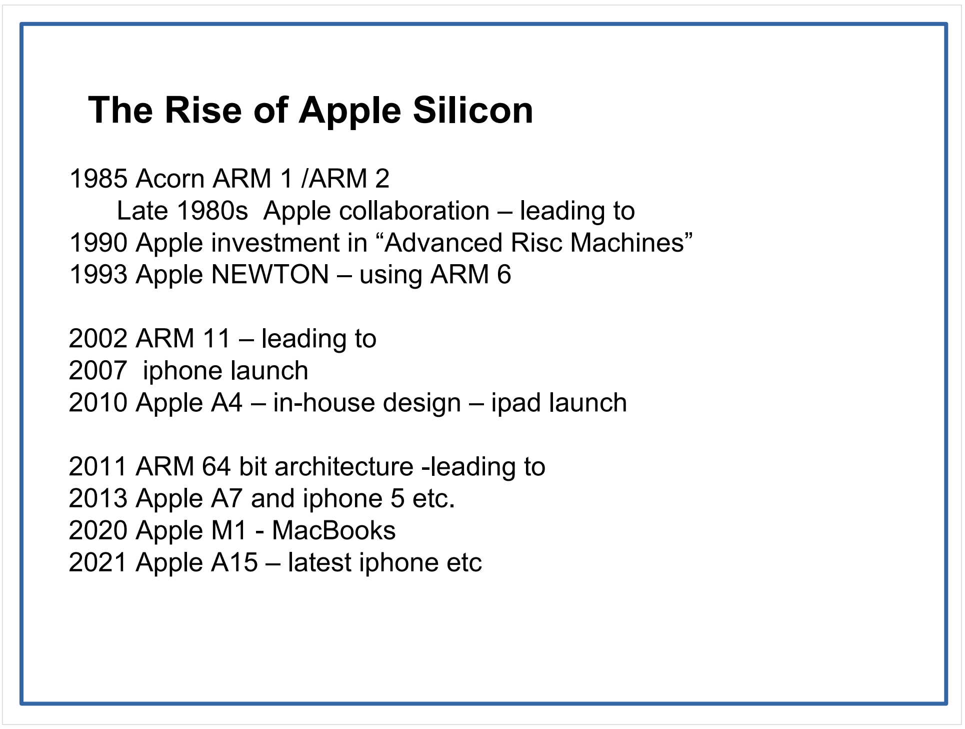
1983 Designed the ARM Instruction Set; emulated the ARM1 on a BBC Micro prior to the making of a chip which worked first time.

(leading to the huge success of ARM processors)

"Sophie nee Roger Wilson, the one-woman once-man whirlwind"

(now "Chief Architect", Broadcom DSL)

Sophie Wilson had a working processor a year later. That was the ARM 1, leading to ARM 2 which was the heart of Acorn's new machines launched in 1987. It had just 30,000 transistors – sounds a lot until you recall the 16 billion figure for its successor 25 years later.



The Rise of Apple Silicon

1985 Acorn ARM 1 /ARM 2
    Late 1980s Apple collaboration – leading to
1990 Apple investment in “Advanced Risc Machines”
1993 Apple NEWTON – using ARM 6

2002 ARM 11 – leading to
2007 iphone launch
2010 Apple A4 – in-house design – ipad launch

2011 ARM 64 bit architecture -leading to
2013 Apple A7 and iphone 5 etc.
2020 Apple M1 - MacBooks
2021 Apple A15 – latest iphone etc

Apple soon became interested in the ARM processor and used it in the Apple Newton, an early tablet. That was a failure – and Apple were then in the doldrums, prior to Steve Jobs return to the Company. But the Newton gave Apple important knowhow.

At the time of Steve Jobs return, Apple Macs were based on PowerPC processors made by Motorola. It was perceived that this was a dead end – that Apple should then switch to Intel, and importantly that Apple should never anchor themselves to one processor architecture. This had some consequences that I’ll return to.

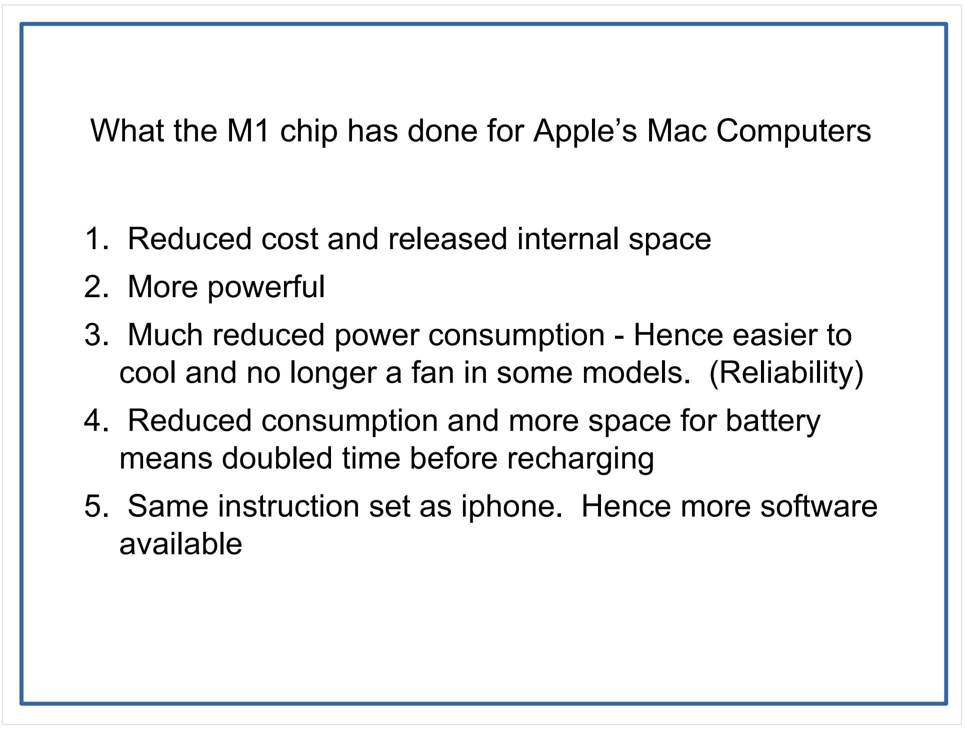
A few years later, development of the iphone went hand in hand in parallel with the evolution of “Apple Silicon”. The first iphone was launched in 2007 with an ARM based chip which now looks very primitive. Apple’s early chips were a collaboration with Samsung and used a 90nm process, that defining the smallest element which could be created on the silicon.

In the next few years Apple designed more and more system-on-chip devices which were at the heart of the ipod, the iphone and the ipad, all ARM based, with a common instruction set.

However Apple’s Mac computers and laptops continued to be Intel based for nearly 10 more years.

But from about 2010 Apple were betting on evolving ARM technology, and it was soon not ‘if’ but ‘when’ they would abandon Intel. They stayed with Intel for many more years, but were progressively developing schemes which would facilitate a change over.

The 90nm process of 2007 had become a 5nm process by 2020 and the Apple M1 arrived as the heart of the newest Macs – with extraordinary benefits.   Apple reduce their production costs – some say by 2 billion dollars a year, if the M1 costs $50 to produce.



What the M1 chip has done for Apple’s Mac Computers

1. Reduced cost and released internal space
2. More powerful
3. Much reduced power consumption - Hence easier to cool and no longer a fan in some models. (Reliability)
4. Reduced consumption and more space for battery means doubled time before recharging
5. Same instruction set as iphone. Hence more software available

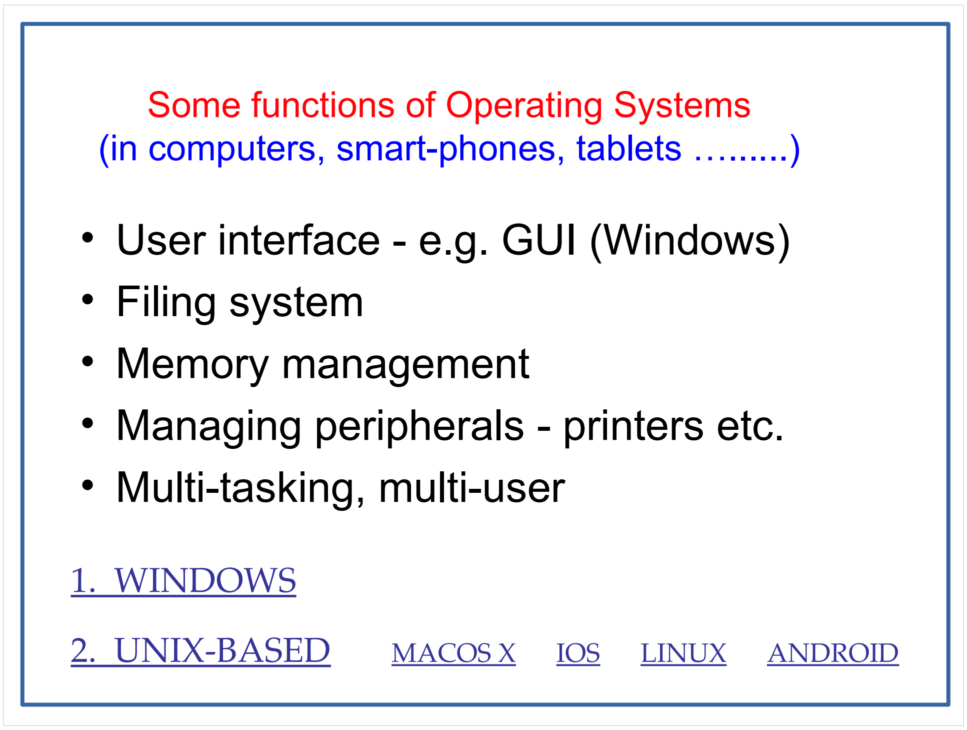
And the Mac computers using the new chip suddenly became much better than their Intel based versions.

More powerful; lower power consumption; cooler (no fan in Macbook air); more internal space for battery; same instruction set as iphones, so much more software; doubled battery life

Apple were not the only ones to choose ARM. ARM ‘s conquests have been surprisingly diverse. There’s the Fugaku supercomputer with some 150,000 ARM processors. There’s the Raspberry Pi. There are very cheap micro-controllers such as the many in modern cars, which are currently in short supply.

Some 180 billion ARM chips have been fabricated,

Last year the Financial Times published an article “Where did it go wrong for Intel”. Intel had chances, but had made bad decisions,


Some functions of Operating Systems
(in computers, smart-phones, tablets ....)

- User interface - e.g. GUI (Windows)
- Filing system
- Memory management
- Managing peripherals - printers etc.
- Multi-tasking, multi-user

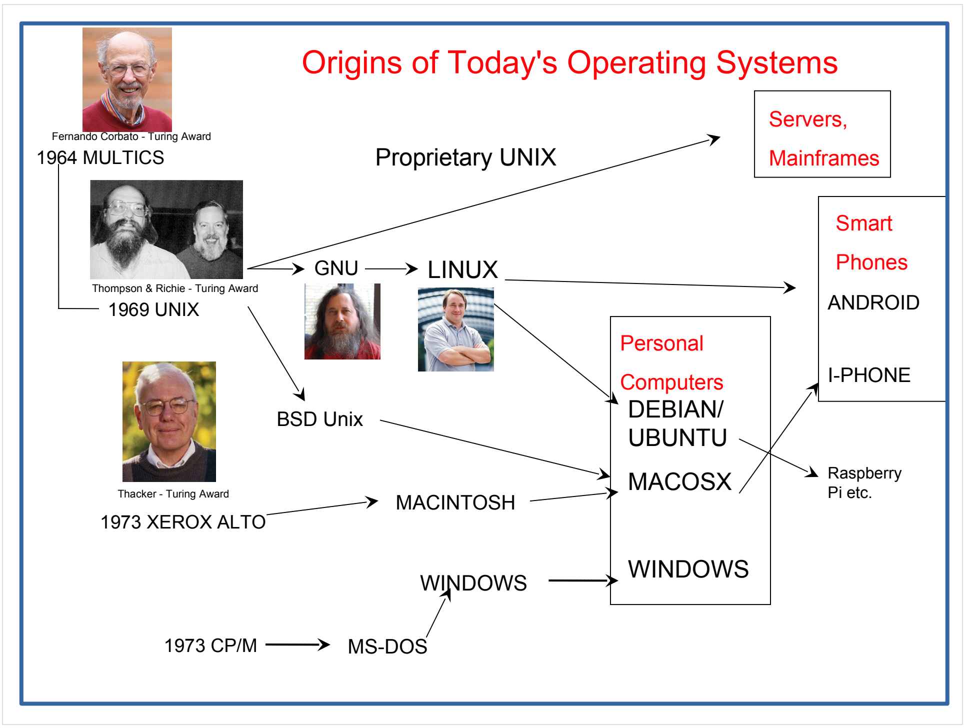
1. WINDOWS

2. UNIX-BASED   MACOS X   IOS   LINUX   ANDROID

Now for the second section of this talk

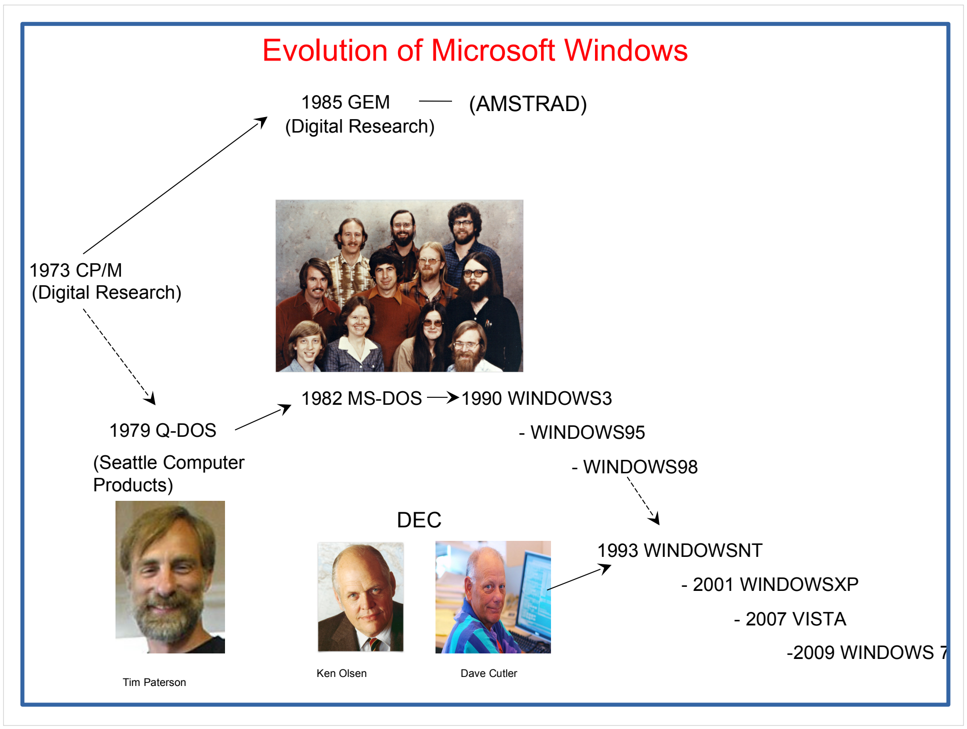
Chips are no good without software, so I'll now turn to operating systems,.

What do operating systems do (from slide)



Early operating systems were created by the computer manufacturer, like IBM.

But in 1964, Fernando Corbato, later a Turing award winner and others began a very ambitious system called Multics. Multics itself was not very successful, but it was the seed for modern systems. The Bell Labs people withdrew from Multics and developed Unix the name being a play on words. (Multics has too many ickses, so lets have one ix.) Unix is still the basis for most modern systems, large and small - from mainframes to today's phones, with all their vast arrays of features. Windows, at the bottom of this diagram, is an exception, so let's just deal with that before returning to the Unix based systems including the Apple Macos which is my main focus.

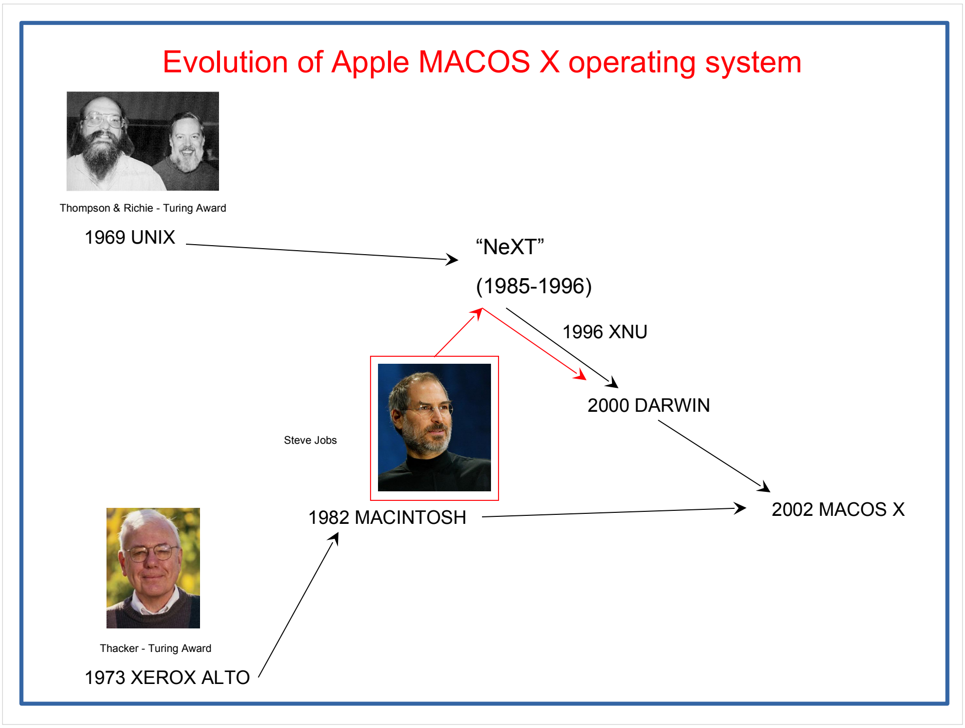


In the early days unix and other operating systems could only run on large systems, and microprocessors were arriving - and CP/M Control Program for Microprocessors was developed in 1973 by Digital Research. This was very successful for some years. Moving on to 1982, IBM were looking for an operating system for the forthcoming IBM PC and talks with Digital Research began - and then collapsed. IBM then gave the contract to the fast-talking Microsoft. Here they are – Bill Gates right there.

Microsoft then had to conjure up an operating system from somewhere - they didn’t have one. So they bought a version of CP/M from Seattle Computer Products for some $50k. They tweaked it to create MS/DOS. They didn't mention the IBM connection, and later Seattle Products sued them, and got a $million settlement.

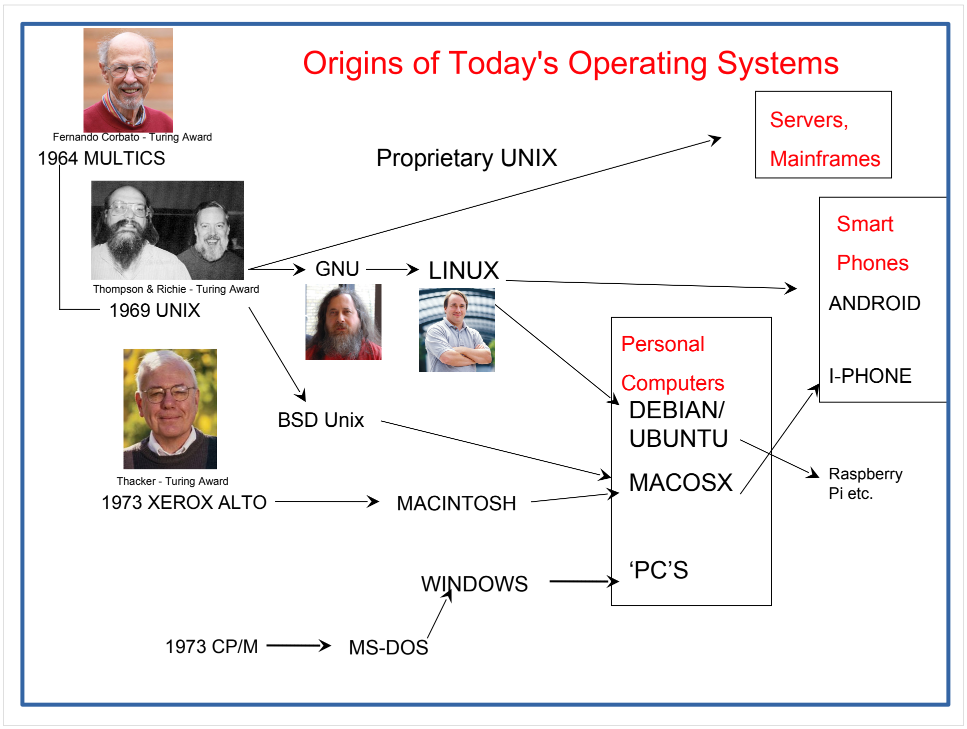
So the IBM PC arrived. The IBM PC world stayed with MS-DOS for a long time. But elsewhere there were glimpses of the future. Right back in 1973, Chuck Thacker, who eventually received a Turing Award, inspired the Xerox Alto, which had the first GUI - Windows-like interface. But it was too far ahead of its time, and we had to wait until the 1980s for the first Apple Macintosh, and rather later, Microsoft Windows. For 10 years, the Microsoft Windows systems were based on DOS foundations. DOS was eventually abandoned with the arrival of Windows XP and its successors.

Just as with DOS, Microsoft had to bring in talent from outside in creating Windows NT. That talent came from Dave Cutler, recruited from the struggling Digital Equipment Corporation, DEC. DEC had great operating systems, created by Cutler; so he and they were the foundation for the new generations of Windows.

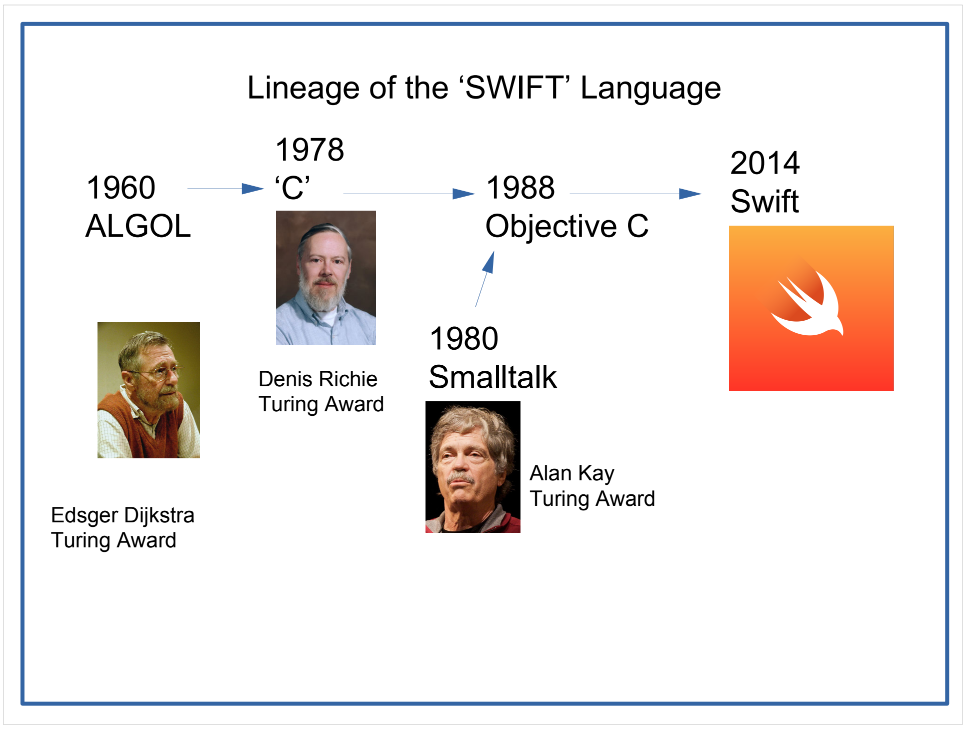


Now back to Apple. They were the first to pick up Thacker's ideas embodied in the revolutionary Xerox Alto. They produced the original Macintosh. But it was still too early, and Apple were in the doldrums in the late 1980s, failing to capitalise on the Macintosh. So Steve Jobs was pushed out in 1985. He formed a company called NeXT with ambitious plans. By 1996 he had a Unix-based operating system called Darwin. That was based on BSD unix, another important product of the University of California, Berkeley. Berkeley had had legal struggles with Bell Labs over the ownership of unix. So Darwin was made unix-like, but not unix. The kernel was called XNU standing for X is not Unix

Steve Jobs then rejoined the still-struggling Apple taking NeXT with him and used Darwin to create the MacOSX operating system which has been a major ingredient of Apple's success. Apple's iOS is another branch with roots in BSD unix



Lets go back to the Unix story in a bit more detail. As I've said, Unix was a proprietary, paid-for system owned by Bell Labs. But in 1983 Richard Stallman came on the scene, crusading against software copyrights. He started GNU, and then the open-software foundation. His original objective was to create a free GNU operating system - Unix-like but not Unix. Hence the acronym GNU – gnu's not Unix, just like XNU. GNU's own developments struggled, but Torvalds came along with Linux in 1991. A number of different groups took up GNU/Linux. Android, and most Raspberry pi operating systems derive from GNU/Linux.


Lineage of the 'SWIFT' Language
1960 ALGOL
1978 'C'
1988 Objective C
1980 Smalltalk
2014 Swift
Edsger Dijkstra Turing Award
Denis Richie Turing Award
Alan Kay Turing Award

Now these days Apple's own software is written using the Apple developed language Swift, improving on Objective C. Swift has had widespread adoption. So lets see where that comes from, and also mention other important languages.

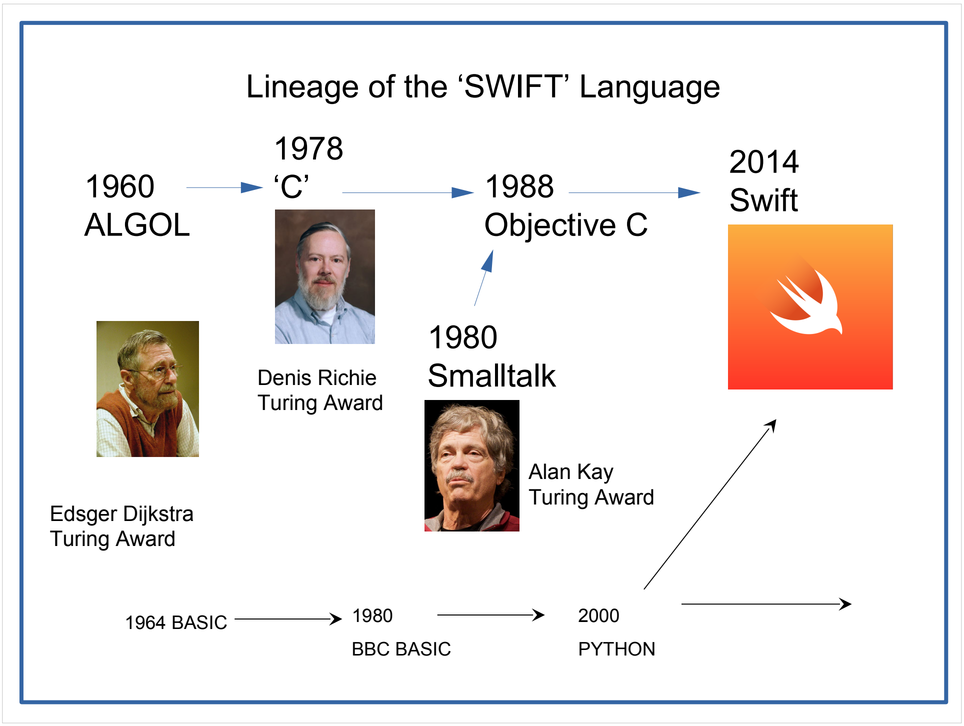
Back in the 1950s it was realised that the computer itself must do some of the work of turning a description of a problem into the machine-code.

The first high-level language was Fortran developed by John Backus at IBM. It was first delivered to users in 1957. Cobol came next, a language designed by committee in 1959 in the USA. Both these languages were very successful in themselves because they were the languages backed by the large vendors like IBM; but they were dead ends. A far more technically significant early language was Algol, which resulted from a joint European/American committee which first met in 1958. Djikstra was one of the main contributors and received a Turing Award for this and other work This led to the language called Algol60. Tony Hoare was another major contributor which earned him a Turing Award. He created the version called Elliott Algol which I learnt and used in 1965. Hoare said of Algol60

ALGOL 60

"Here is a language so far ahead of its time, that it was not only an improvement on its predecessors, but also on nearly all its successors"

(Tony Hoare – 1980 Turing Award)



Not many people used Algol, compared with Fortran and Cobol. But Algol led first to Coral66, Pascal (1970), then C (1972) and C++ (1983) Objective C and then Swift. Using these compiled languages was a two stage process. Stage one was compilation, where the natural language instruction were turned by the computer into machine code suitable for that specific computer, and stage two was to run the generated instructions.

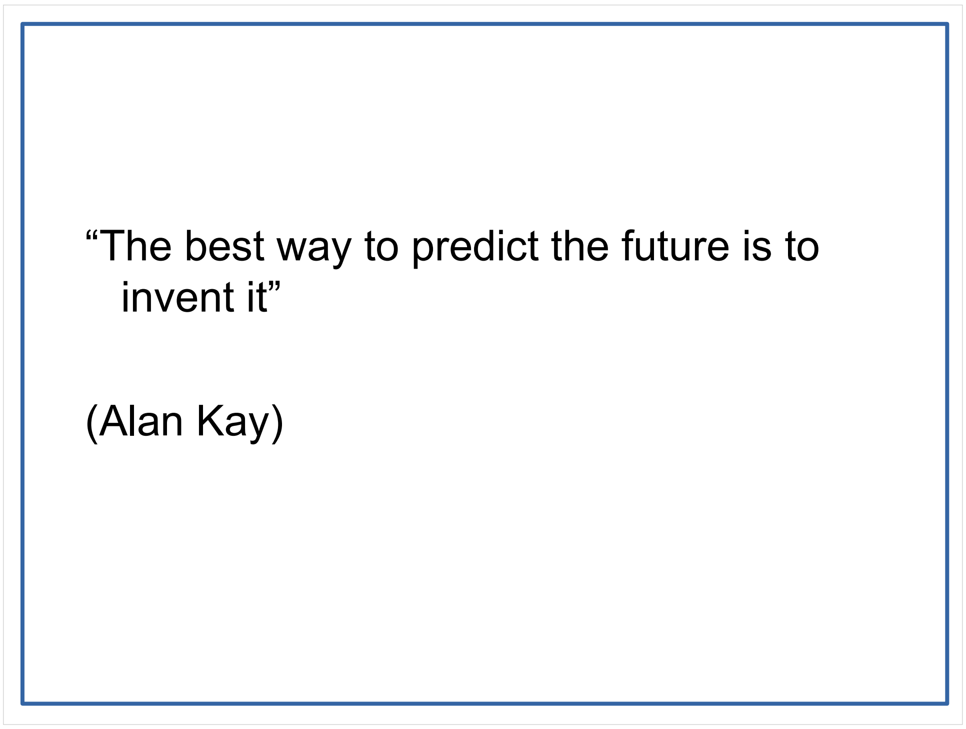
In the late 1960s most programs were still fed to mainframe machines via cardreaders, and run in succession; it was all pretty tedious. But it was becoming possible for a user at a terminal to work directly more nearly as we do now. This was the stimulus for BASIC (1964) a very simple "interpreted" language. This later led to BBC BASIC, usually regarded as the best implementation of BASIC. With interpreted languages, the compilation is part of the run process. It was much less efficient and so the need for compiled languages for professional computing remained.

On now to modern times.

The days have gone when computers in schools and at home had to have BASIC and were provided with a manual on BASIC. But in fact programming facilities are there, either tucked away, or down-loadable, often free of charge.

Versions of C are still pre-eminent for professionals.

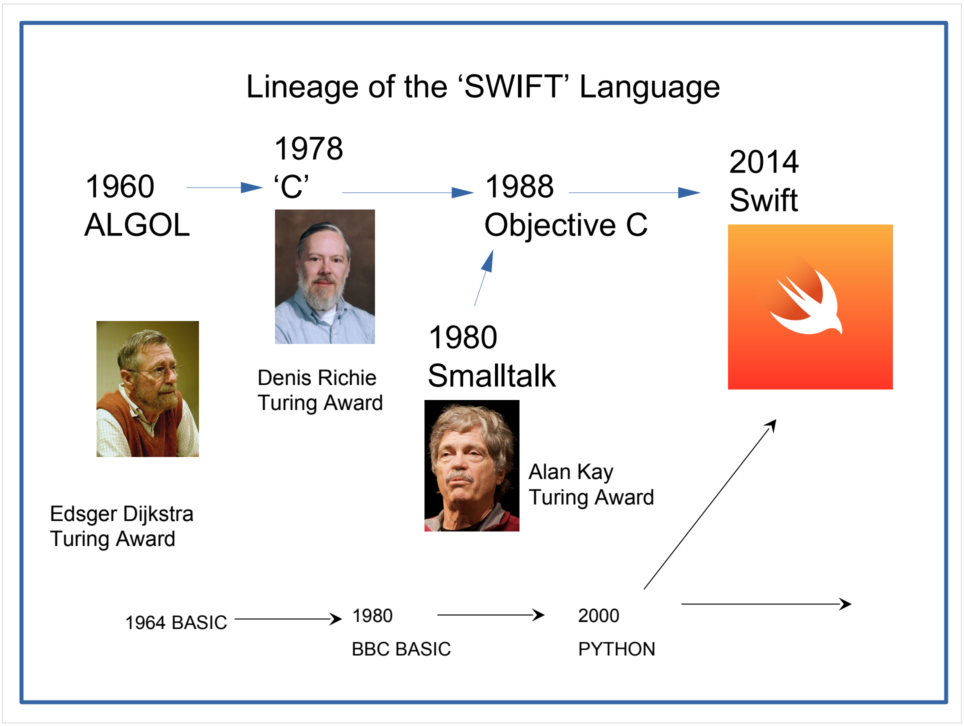
But there are now hundreds of other languages - and there are reasons why many new ones have been developed - some in response to changing circumstances. One thing that has happened is the enormous reduction in the cost of computer hardware. In earlier times it was vital to create compact, efficient, software, so the emphasis was on languages like C generating very efficient code for efficient use of expensive computer hardware. Things changed from the 1980s.



"The best way to predict the future is to invent it"

(Alan Kay)

Alan Kay was a visionary at Xerox Parc working with Chuck Thacker on the pioneering Xerox Alto. He is known for saying that "the best way of predicting the future is to invent it". He devised Smalltalk in 1980, pioneering the Object Oriented style of programming – hugely influential, and feeding through into Swift.



Perl in 1987 was designed "to make efficient use of expensive computer programmers".

The   next thing was the increased importance of software which can run on any computer. A web page for example must be rendered in essentially the same way on any computer. Two particular languages are associated with web pages - HTML (1990) and other "mark-up" languages, and Java (1995).

And then came Python. This could be said to be a successor to BASIC, though much more powerful. It is very popular and easy to run on PCs, Macs, Linux etc - and it is free. Python has become the most popular language

There's another thing about Python. The vast majority of computer languages are in English. Unusually, Python has a chinese version.

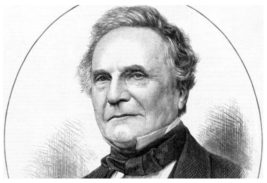
But the C language has moved on. Much of Apple's software was written in Objective C. But objective C was a product of the early 1980s, based on even older ideas but introducing Alan Kay's Smalltalk ideas. So in 2010 Apple decided to develop Swift, based on Objective C but with many new features, including some from Python. Swift was launched in 2014 and has been very successful.

20 years ago Apple made the decision that they should not become anchored to a single processor architecture. They took some ideas on 'Universal Architecture' from NextStep, and Universal Binaries are the result – single pieces of software which can run on two different processors, currently the Intel or the M1. Their other investment was into Rosetta, a very efficient translator which can convert Intel code to M1 code with very little degradation in performance.

From 1821 -

It was all Babbage’s Idea !

“..to calculate by steam”

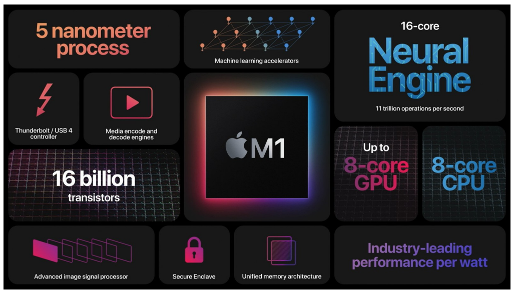


To 2021

- the Apple

M1

System on Chip


5 nanometer process
Machine learning accelerators
16-core
Neural Engine
11 trillion operations per second
Thunderbolt / USB 4 controller
Media encode and decode engines
M1
16 billion transistors
Up to
8-core GPU
8-core CPU
Advanced image signal processor
Secure Enclave
Unified memory architecture
Industry-leading performance per watt

Wouldn’t Babbage have been amazed.